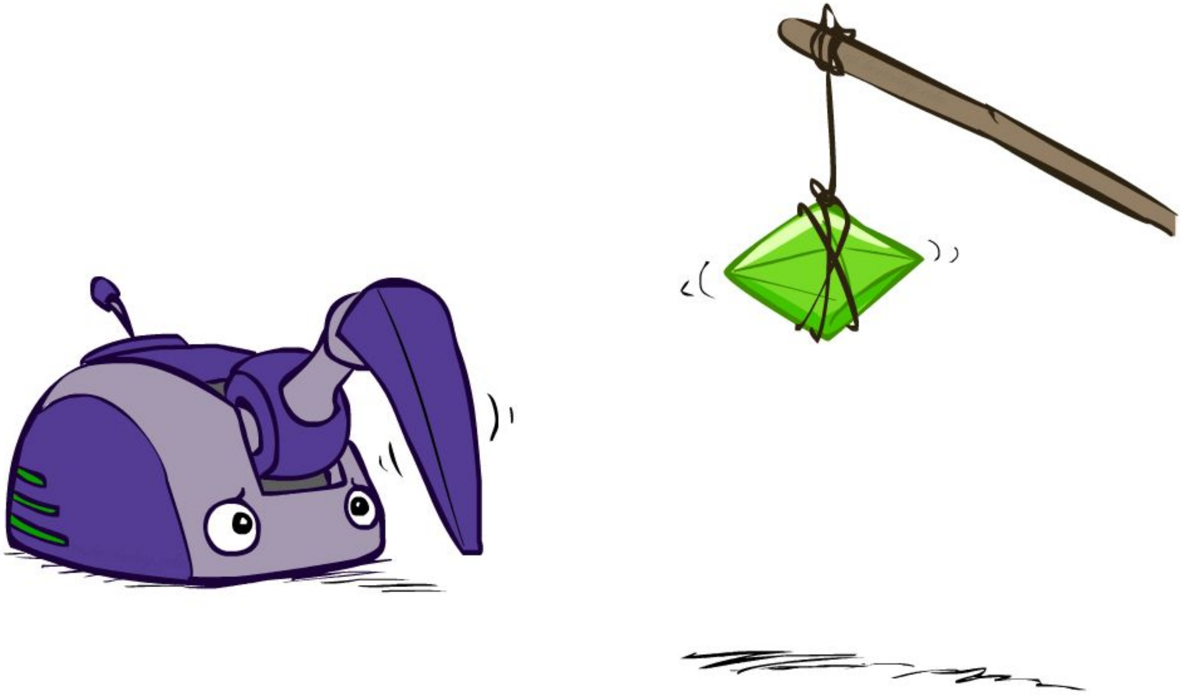
Most materials in this chapter are based on materials from http://aima.cs.berkeley.edu/ and http://ai.berkeley.edu

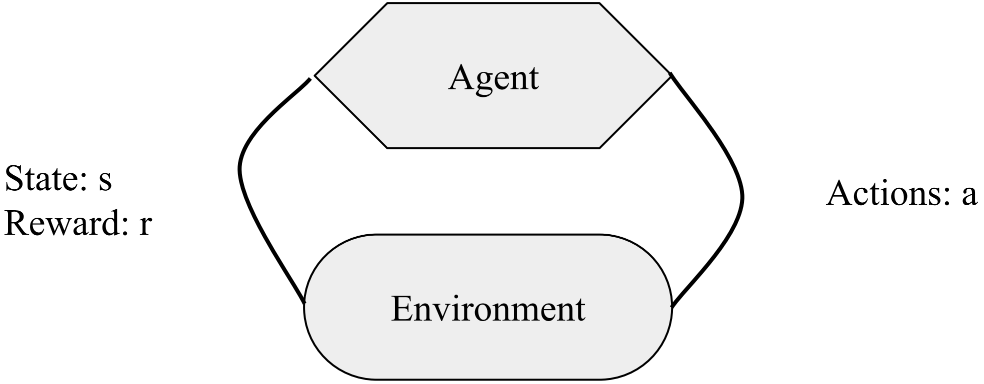
3. Reinforcement Learning

COMP 3270
Artificial Intelligence

Dirk Schnieders



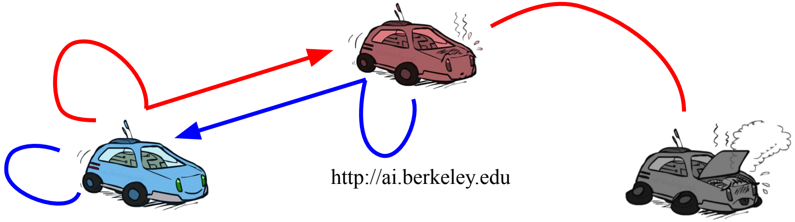
Reinforcement Learning - Idea



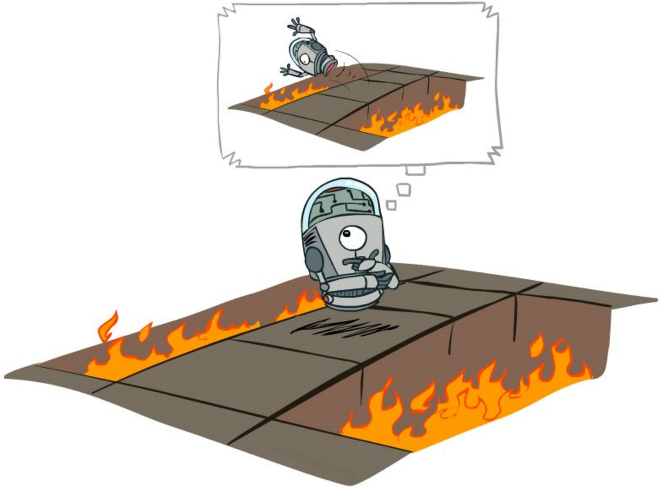
- Basic idea:
  - Receive feedback in the form of rewards
  - Agent’s utility is defined by the reward function
  - Must (learn to) act so as to maximize expected rewards
  - All learning is based on observed samples of outcomes!

Reinforcement Learning

- It’s just the same ...
  - Still assume a Markov decision process (MDP):
    - A set of states $s \in S$
    - A set of actions (per state) A
    - A model $T(s,a,s')$
    - A reward function $R(s,a,s')$
  - Still looking for a policy $\pi(s)$
- New
  - Don’t know T or R
    - I.e., we don’t know what the actions do
  - Must actually try actions and states out to learn



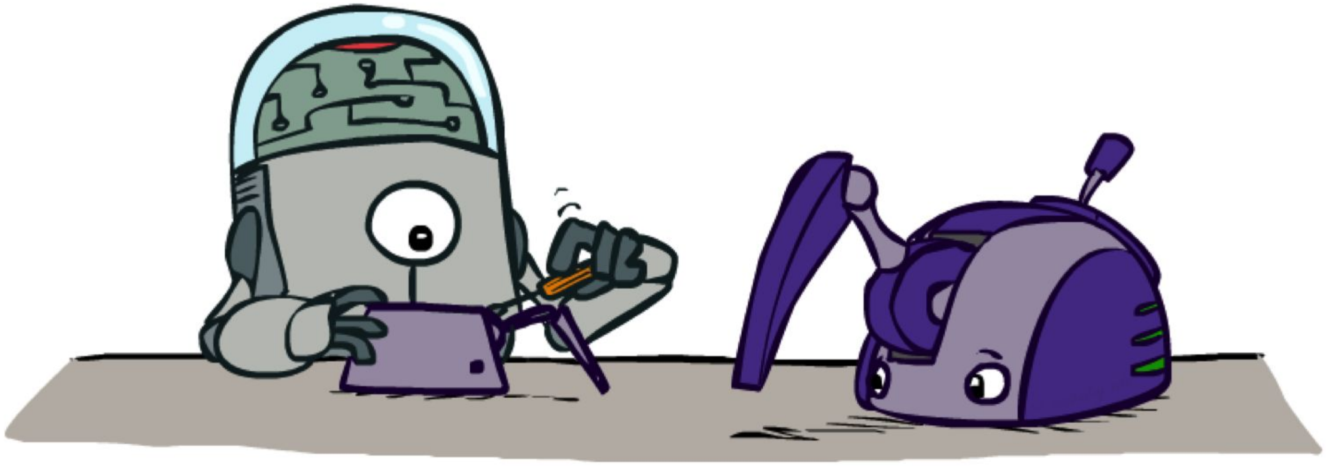
Offline (MDPs) vs. Online (RL)



Offline

Online

Model-Based Learning



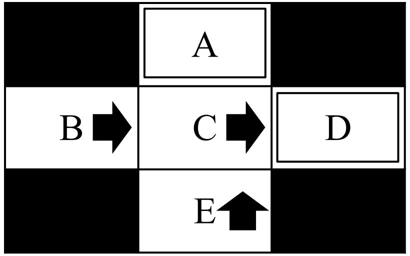
Model-Based Learning

- Model-Based Idea:
  - Learn an approximate model based on experiences
  - Solve for values as if the learned model were correct
  - Step 1: Learn empirical MDP model
    - Count outcomes s' for each s, a
    - Normalize to give an estimate of T^(s, a, s')
    - Discover each R^(s, a, s') when we experience (s, a, s')
  - Step 2: Solve the learned MDP
    - For example, use policy iteration, as before

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} \hat{T}(s, \pi_i(s), s')[\hat{R}(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')]$$

Example: Model-Based Learning

Given: π



$\gamma = 1.0$

Episode 1:
B, east, C, -1
C, east, D, -1
D, exit, , +10

Episode 2:
B, east, C, -1
C, east, D, -1
D, exit, , +10

Episode 3:
B, east, C, -1
C, east, D, -1
D, exit, , +10

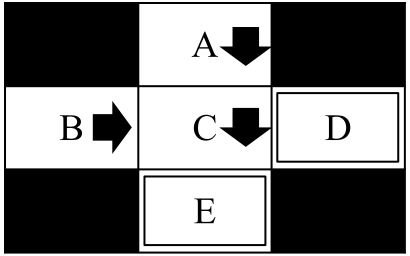
Episode 4:
E, north, C, -1
C, east, A, -1
A, exit, , -10

T^(s, a, s'):
T^(B, east, C) = 1.00
T^(C, east, D) = 0.75
T^(C, east, A) = 0.25
...

R^(s, a, s'):
R^(B, east, C) = -1
R^(C, east, D) = -1
R^(C, east, A) = -1
...

Quiz - Model-Based Learning

Given: $\boldsymbol{\pi}$



$\gamma = 1.0$

Episode 1:
A, south, C, -1
C, south, E, -1
E, exit, , 10

Episode 2:
B, east, C, -1
C, south, D, -1
D, exit, , -10

Episode 3:
B, east, C, -1
C, south, E, -1
E, exit, , 10

Episode 4:
A, south, C, -1
C, south, E, -1
E, exit, , 10

$\hat{T}$(s, a, s'):
$\hat{T}$(A, south, C) = ?
$\hat{T}$(B, east, C) = ?
$\hat{T}$(C, south, E) = ?
$\hat{T}$(C, south, D) = ?

Example: Expected Age

- Goal: Compute expected age of comp3270 students

Known P(a)
$E[A] = \sum_a P(a) \cdot a \quad = 0.35 \times 20 + \ldots$

Unknown P(a): Model based
$\hat{P}(a) = \frac{\text{num}(a)}{N}$ $E[A] \approx \sum_a \hat{P}(a) \cdot a$

Unknown P(a): Model free
$E[A] \approx \frac{1}{N} \sum_i a_i$

Quiz - Model-Based Learning

- Consider an MDP with 3 states, A, B and C; and 2 actions CW and CCW
- We are given samples of what an agent experiences
- In this problem, we will first estimate the model
    - i.e., the transition function and the reward function
- Consider the samples (shown on the next slide) that the agent encountered

Quiz (continued) - Model-Based Learning

s	a	s'	r
A	CW	C	-9.0
A	CW	C	-9.0
A	CW	B	0.0
A	CW	B	0.0
A	CW	B	0.0
A	CCW	B	0.0
A	CCW	B	0.0
A	CCW	B	0.0
A	CCW	C	-3.0
A	CCW	B	0.0

s	a	s'	r
B	CW	A	2.0
B	CW	A	2.0
B	CW	A	2.0
B	CW	A	2.0
B	CW	A	2.0
B	CCW	A	-6.0
B	CCW	A	-6.0
B	CCW	C	5.0
B	CCW	C	5.0
B	CCW	C	5.0

s	a	s'	r
C	CW	A	3.0
C	CW	A	3.0
C	CW	A	3.0
C	CW	A	3.0
C	CW	B	-4.0
C	CCW	A	0.0
C	CCW	A	0.0
C	CCW	A	0.0
C	CCW	A	0.0
C	CCW	A	0.0

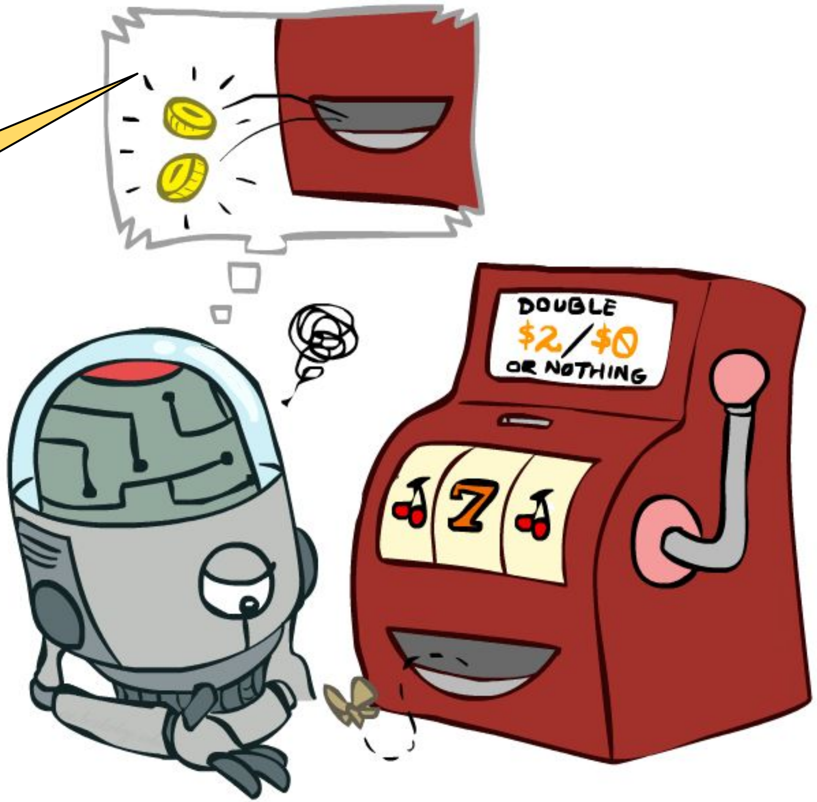
Quiz (continued) - Model-Based Learning

- Estimating the transition function, T(s,a,s') and reward function R(s,a,s') for this MDP
- Fill in the missing values in the table for T(s,a,s') and R(s,a,s')

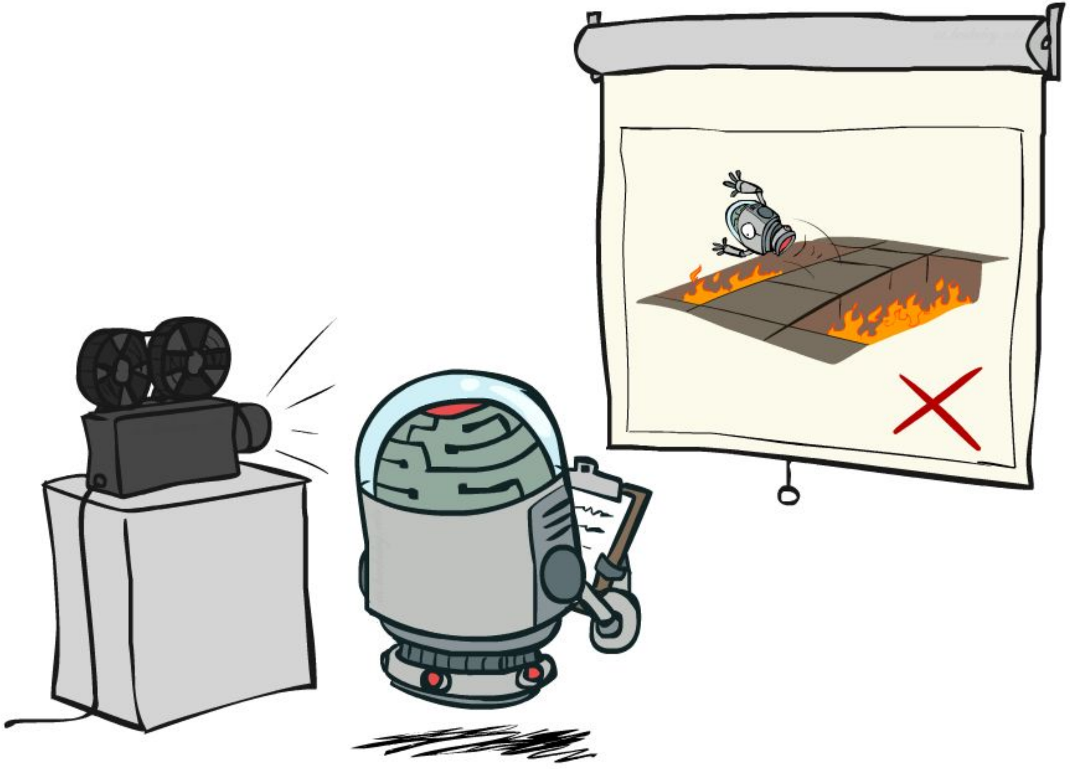
s	a	s'	T(s,a,s')	R(s,a,s')
A	CW	B		
A	CW	C		
A	CCW	B	0.8	0.0
A	CCW	C	0.2	-3.0
B	CW	A	1.0	2.0
B	CCW	A	0.4	-6.0
B	CCW	C	0.6	5.0
C	CW	A	0.8	3.0
C	CW	B	0.2	-4.0
C	CCW	A	1.0	0.0

Model-Free Learning

In model-free learning you compare what you want vs. what you got



Passive Reinforcement Learning



Passive Reinforcement Learning

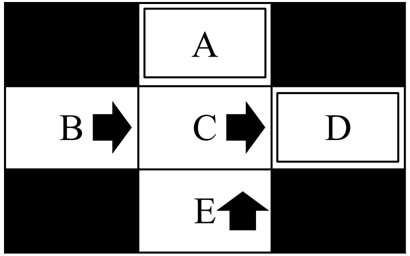
- Simplified task: policy evaluation
  - Input: a fixed policy $\pi(s)$
  - You don’t know the transitions T(s,a,s’)
  - You don’t know the rewards R(s,a,s’)
  - Goal: learn the state values
- In this case:
  - Learner is “along for the ride”
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - NOT offline planning! You actually take actions in the world

Direct Evaluation

- Goal: Compute values for each state under π
- Idea: Average together observed sample values
  - Act according to π
  - Every time you visit a state, write down what the sum of discounted rewards turned out to be
  - Average those samples
- This is called direct evaluation

Example: Direct Evaluation

Given: π



$\gamma = 1.0$

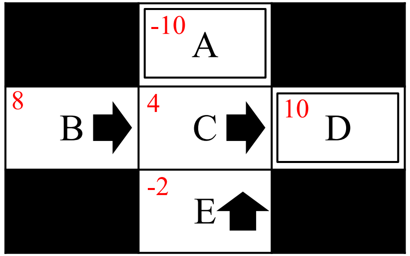
Episode 1:
B, east, C, -1
C, east, D, -1
D, exit, , +10

Episode 2:
B, east, C, -1
C, east, D, -1
D, exit, , +10

Episode 3:
E, north, C, -1
C, east, D, -1
D, exit, , +10

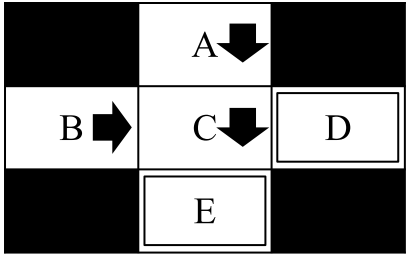
Episode 4:
E, north, C, -1
C, east, A, -1
A, exit, , -10

Output: Values



Quiz - Direct Evaluation

Given: π



$\gamma = 1.0$

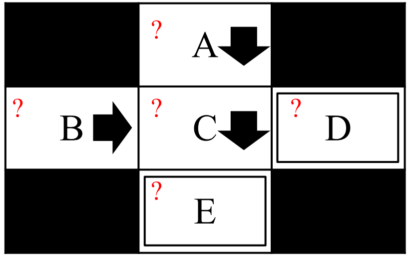
Episode 1:
A, south, C, -1
C, south, E, -1
E, exit, , +10

Episode 2:
B, east, C, -1
C, south, D, -1
D, exit, , -10

Episode 3:
B, east, C, -1
C, south, E, -1
E, exit, , +10

Episode 4:
A, south, C, -1
C, south, E, -1
E, exit, , +10

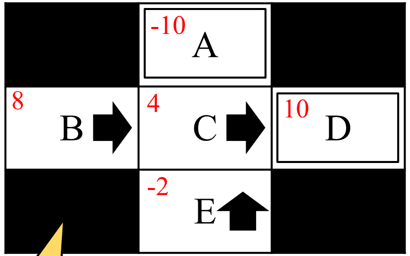
Output: Values



Problems with Direct Evaluation

- Positives
  - Easy to understand
  - Knowledge of T and R not required
  - Eventually computes the correct values
- Negatives
  - State connections are not considered
  - Each state must be learned separately
  - Takes a long time to learn

Output: Values



B and E both go to C under this policy. However, their values are very different ...

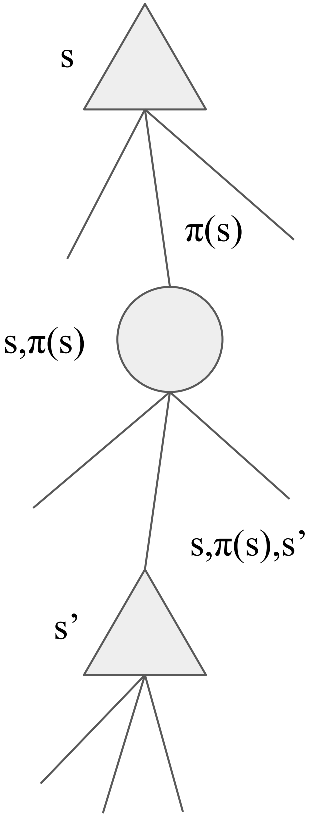
Why Not Use Policy Evaluation?

- Can we use policy evaluation?
  - Each round, replace V with a one-step-look-ahead layer over V

$$V_0^\pi(s) = 0$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^\pi(s')]$$

- This approach fully exploits the connections
  - Unfortunately, we need T and R to do it



Sample-Based Policy Evaluation?

- We want to improve our estimate of V by computing these averages:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]$$

- Idea: Take samples of outcomes s' (by doing the action!) and average

$$sample_1 = R(s,\pi(s),s_1') + \gamma V_k^{\pi}(s_1')$$
$$sample_2 = R(s,\pi(s),s_2') + \gamma V_k^{\pi}(s_2')$$
$$\ldots$$
$$sample_n = R(s,\pi(s),s_n') + \gamma V_k^{\pi}(s_n')$$

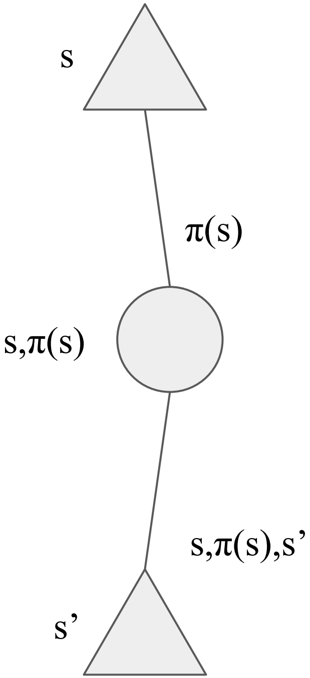
Almost, however we cannot rewind time to get sample after sample from state s

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n}\sum_i sample_i$$

Will this work?

Temporal Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often
- Temporal difference learning of values
  - Policy still fixed, still doing evaluation!
  - Move values toward value of whatever successor occurs: running average



Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

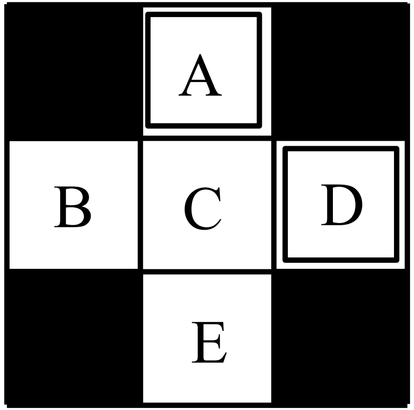
Temporal Difference Learning

- Exponential Moving Average
  - The running interpolation update makes recent samples more important
  - Forgets about the past (distant past values were wrong anyway)
- Decreasing learning rate (alpha) can give converging averages

Demo

Another Example

States

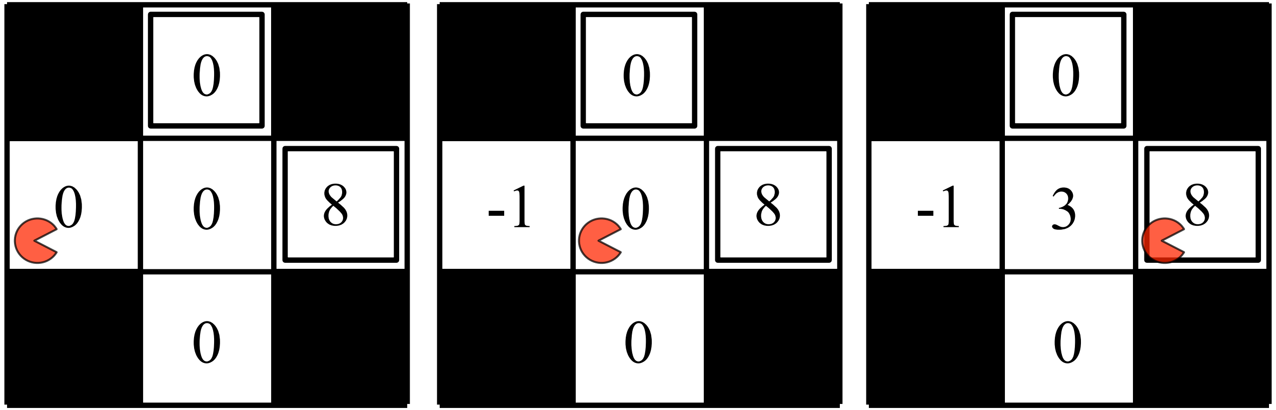


$\alpha = 0.5, \gamma = 1.0$

Observed Transitions

B, east, C, -2

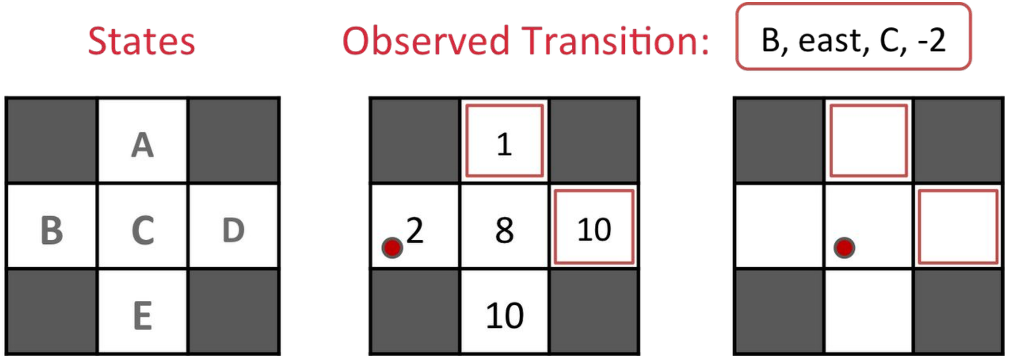
C, east, D, -2



$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

Quiz - Temporal Difference Learning

- Consider the grid world shown below
- The left panel shows the name of each state A through E. The middle panel shows the current estimate of the value function V^π for each state
- A transition is observed, that takes the agent from state B through taking action east into state C, and the agent receives a reward of -2.
- What are the value estimates after the TD learning update?



Assume: $\gamma = 1$, $\alpha = 1/2$

$$V^\pi(s) \leftarrow (1-\alpha)V^\pi(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^\pi(s')\right]$$

TD Value Learning

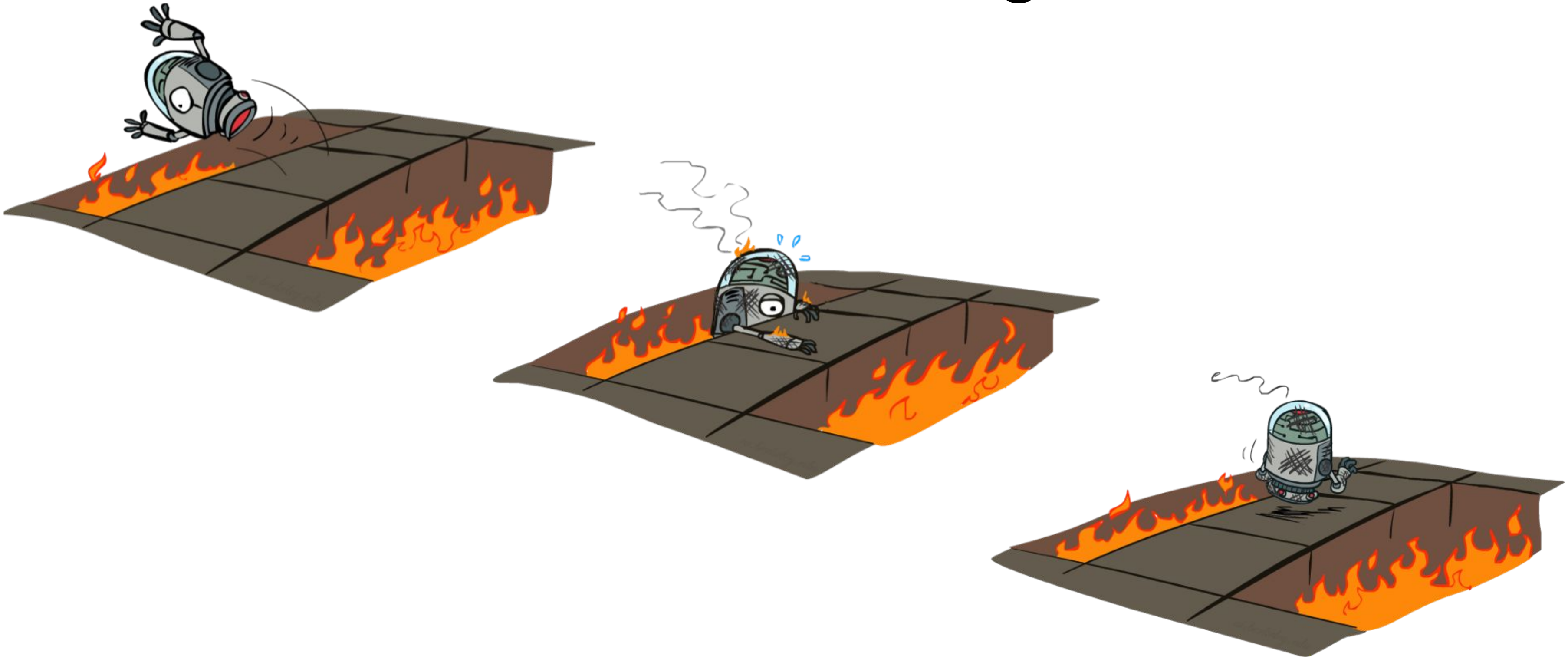
- Model-free way to do policy evaluation, mimicking Bellman updates with running sample averages
- However, we cannot turn values into a new policy

$$\pi(s) = \arg\max_a Q(s,a)$$

$$Q(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V(s')\right]$$

- Idea
  - Learn Q-values, not values
  - Makes action selection model-free

Active Reinforcement Learning



Detour: Q-Value Iteration

- Value iteration: find successive (depth-limited) values
  - Start with $V_0(s) = 0$, which we know is right
  - Given V_k, calculate the depth k+1 values for all states:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

- But Q-values are more useful, so compute them instead
  - Start with $Q_0(s,a) = 0$, which we know is right
  - Given Q_k, calculate the depth k+1 q-values for all q-states:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')\right]$$

Q-Learning

- Q-Learning: sample-based Q-value iteration

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$$

- Learn Q(s,a) values as you go
  - Receive a sample (s,a,s’,r)
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:

    $sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$
  - Incorporate the new estimate into a running average:

    $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\,[sample]$

Demo

Quiz - Q-Learning

- Consider an MDP with 3 states, A, B and C; and 2 actions CW and CCW
- In this quiz, instead of first estimating the transition and reward functions, we will directly estimate the Q function using Q-learning
- Assume, the discount factor is 0.5 and the step size for Q-learning is 0.5
- Let the current Q function, Q(s,a), be

	A	B	C
CW	-2	2	1
CCW	0.5	2	10

- The agent encounters the following samples

s	a	s'	r
A	CW	B	0.0
B	CW	A	2.0

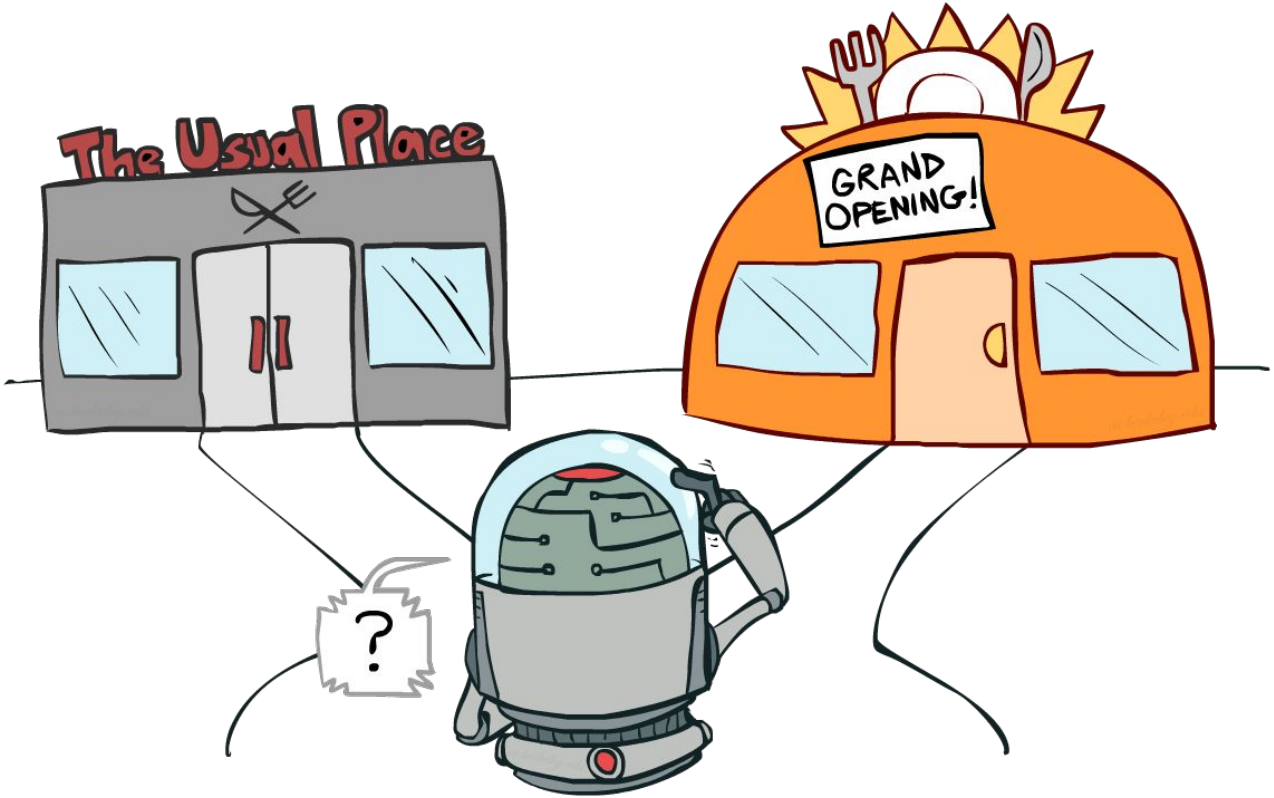
- Process the samples given above and fill in the Q-values after both samples have been accounted for

	A	B	C
CW			
CCW			

Q-Learning Properties

- Amazing result
  - Q-learning converges to optimal policy -- even if you're acting suboptimally!
  - This is called off-policy learning
- Limitations:
  - You have to explore enough
  - You have to eventually make the learning rate small enough
    - … but not decrease it too quickly
  - Basically, in the limit, it doesn't matter how you select actions

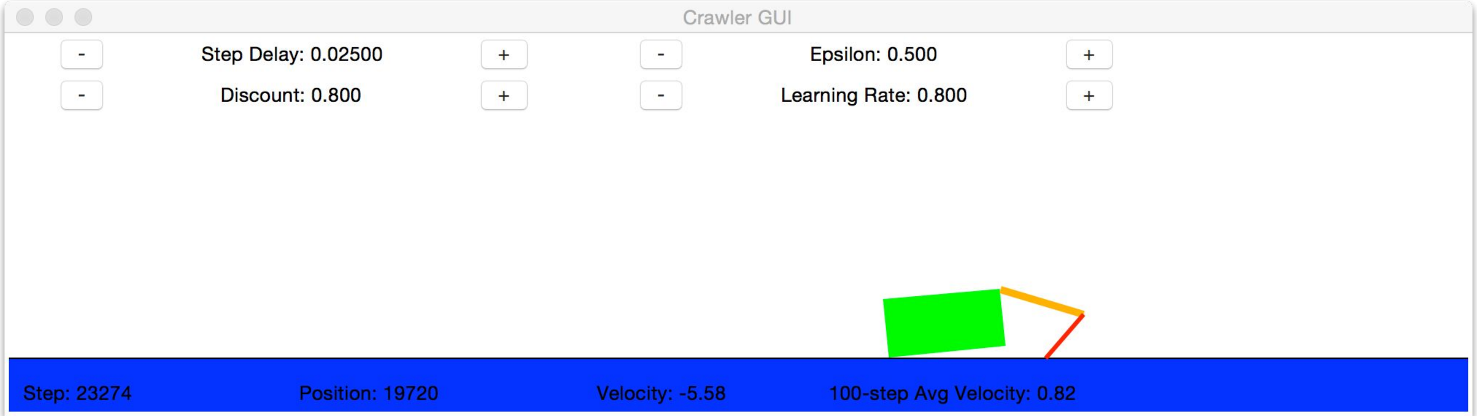
Exploration vs. Exploitation



How to Explore?

- Several schemes for forcing exploration
- Simplest: random actions (ε-greedy)
  - Every time step, flip a coin
    - With (small) probability ε, act randomly
    - With (large) probability 1-ε, act on current policy
  - Problems with random actions?
    - You do eventually explore the space, but keep thrashing around once learning is done
  - Solution
    - Lower ε over time
    - Exploration functions

Crawler in Assignment 3



Exploration Functions

- When to explore?
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established, eventually stop exploring
- Exploration function
  - Takes a value estimate u and a visit count n, and returns an optimistic utility, e.g. $f(u,n) = u + k/n$

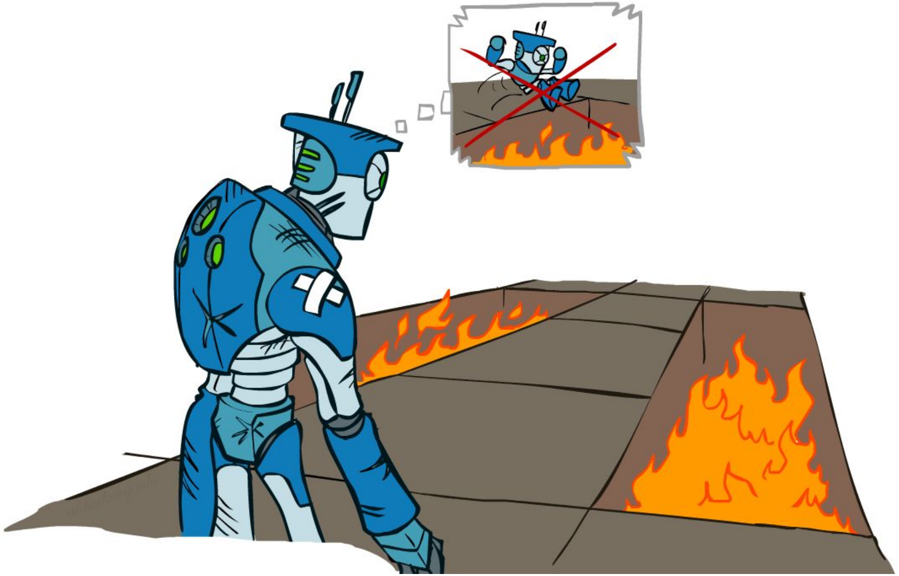
$$Q(s,a) \leftarrow_{\alpha} R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

$$Q(s,a) \leftarrow_{\alpha} R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$$

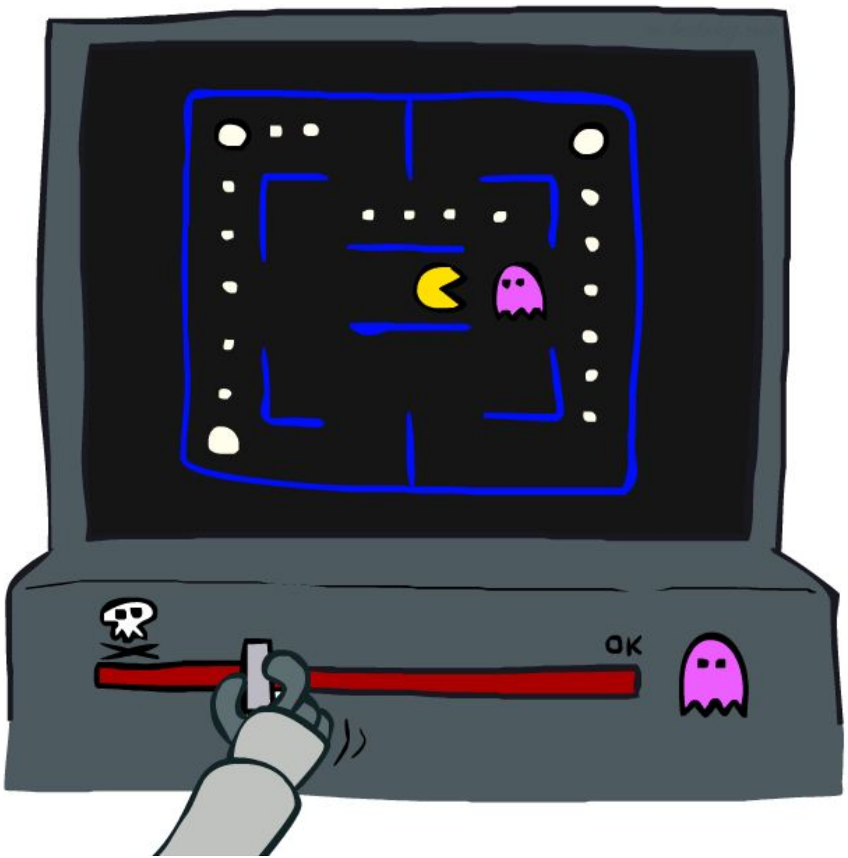
- Note: this propagates the “bonus” (k) back to states that lead to unknown states as well!

Regret

- Even if you learn the optimal policy, you still make mistakes along the way!
- Regret is a measure of your total mistake cost: the difference between your (expected) rewards, including youthful suboptimality, and optimal (expected) rewards
- Minimizing regret goes beyond learning to be optimal – it requires optimally learning to be optimal
- Example: random exploration and exploration functions both end up optimal, but random exploration has higher regret

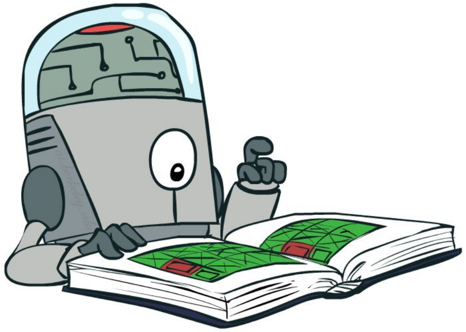


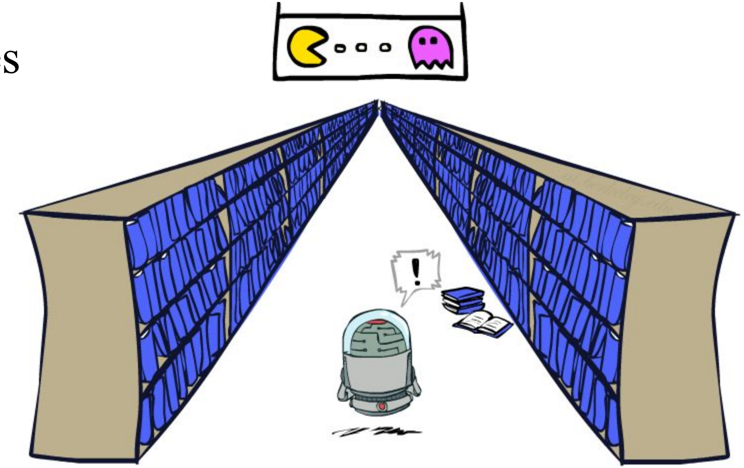
Approximate Q-Learning



Generalizing Across States

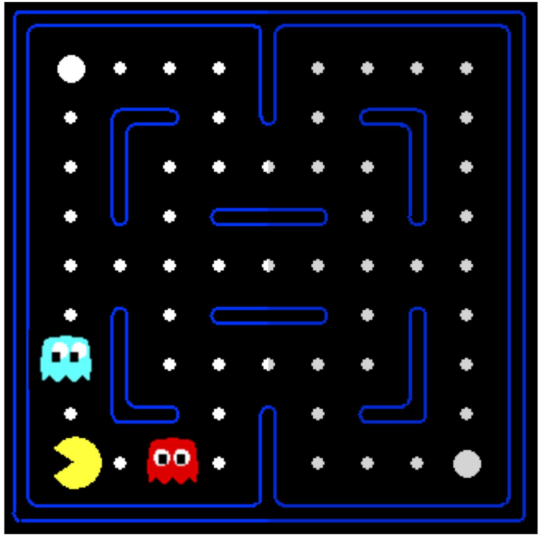
- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory
- Instead, we want to generalize:
  - Learn about some small number of training states from experience
  - Generalize that experience to new, similar situations
  - This is a fundamental idea in machine learning



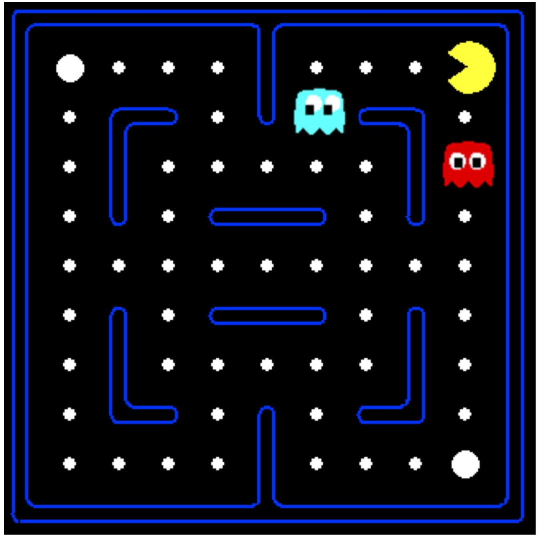


Example: Pacman

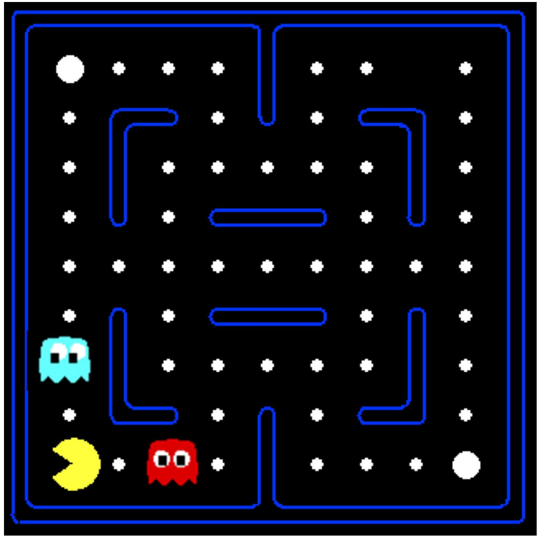
Let's say we discover through experience that this state is bad



In naive q-learning, we know nothing about this state

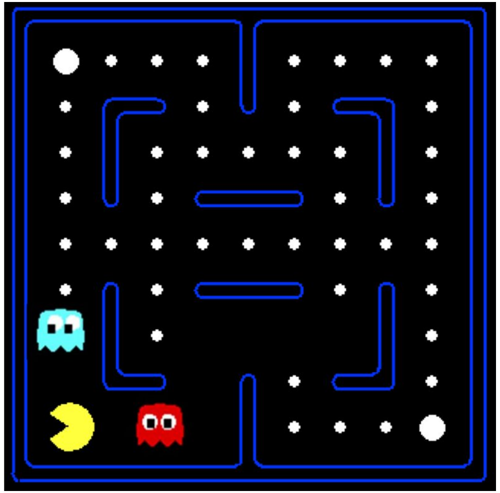


Or even this one!



Feature-Based Representations

- Solution: describe a state using a vector of features (properties)
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
  - Example features:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - 1 / (dist to dot)^2
    - Is Pacman in a tunnel? (0/1)
    - …… etc.
    - Is it the exact state on this slide?



- Can also describe a q-state (s, a) with features (e.g. action moves closer to food)

Linear Value / Q Functions

- Using a feature representation, we can write a Q-function (or value function) for any state using a few weights:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- Advantage
  - Our experience is summed up in a few powerful numbers
- Disadvantage
  - States may share features but actually be very different in value

Approximate Q-Learning

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

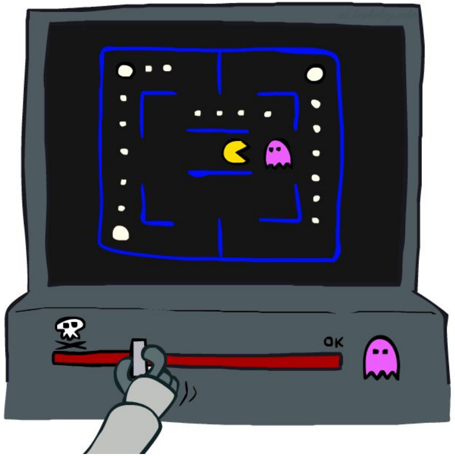
- Q-learning with linear Q-functions

$$\text{transition } = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\, [\text{difference}] \qquad \text{Exact Q's}$$

$$w_i \leftarrow w_i + \alpha\, [\text{difference}]\, f_i(s,a) \qquad \text{Approximate Q's}$$



- Intuitive interpretation:
  - Adjust weights of active features
  - E.g., if something unexpectedly bad happens, blame the features that were on
    - disprefer all states with that state's features

Example: Q-Pacman

$$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s

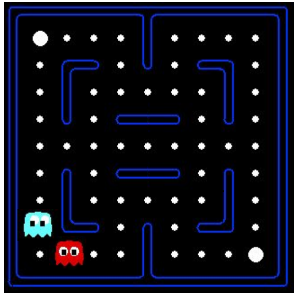
$$f_{DOT}(s, \text{NORTH}) = 0.5$$

$$f_{GST}(s, \text{NORTH}) = 1.0$$

$a = \text{NORTH}$

$r = -500$

s'



$$Q(s, \text{NORTH}) = +1$$

$$Q(s', \cdot) = 0$$

$$r + \gamma \max_{a'} Q(s', a') = -500 + 0$$

$$\text{difference} = -501$$

$$w_{DOT} \leftarrow 4.0 + \alpha \, [-501] \, 0.5$$

$$w_{GST} \leftarrow -1.0 + \alpha \, [-501] \, 1.0$$

$$Q(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$$

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

Quiz: Approx. Q-Learning

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s',a')\right] - Q(s,a)$$

$$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s,a)$$

- Consider an unknown MDP where there are three states [A, B, C] and two actions [1, 2]
- We are given the following samples generated from taking actions
- Assume $\gamma = 1$ and $\alpha = 0.5$ and the following two features
- $f_1(s, a) = 1$ and $f_2(s, a) = (-1)^a$
- Starting from initial weights of 0, compute the updated weights after observing the following two samples

s	a	s’	r
A	2	B	4
B	1	A	0

- What are the weights after the first update?
- What are the weights after the second update?